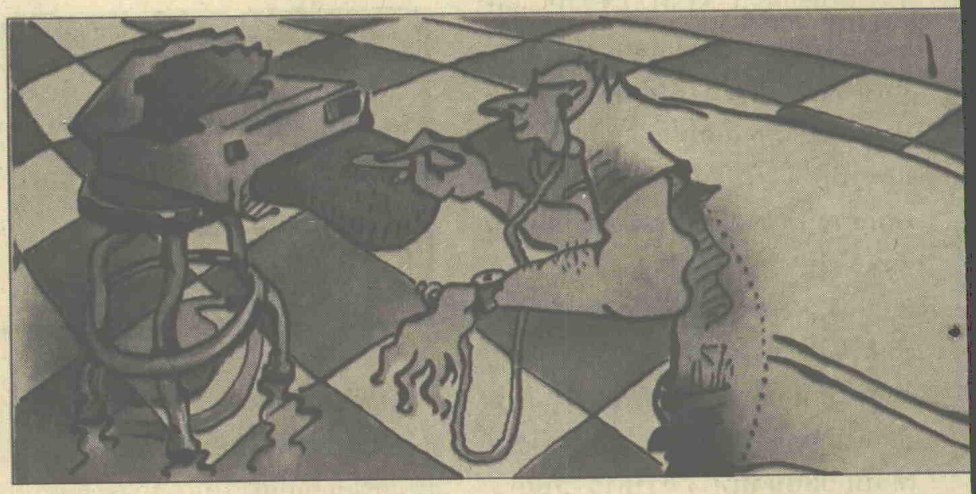
จเร เลิศสุดวิชัย

การสร้าง โอเวอร์เลย์ ให้กับคลิปเปอร์

อธิบายถึงขั้นตอนการทำโอเวอร์เลย์ให้กับโปรแกรมที่พัฒนาด้วยคลิปเปอร์



ปัญหาหนึ่งที่สำคัญของการพัฒนาโปรแกรมด้วยคลิปเปอร์คือโปรแกรมผลลัพธ์ (.EXE) ที่ได้จะมีขนาดใหญ่ ซึ่งหากโปรแกรมมีขนาดใหญ่เกินกว่าหน่วยความจำของเครื่องที่มีอยู่ก็จะทำให้โปรแกรมนั้นไม่สามารถทำงานบนเครื่องได้ การแก้ไขนั้นทำได้โดยแบ่งโปรแกรมที่พัฒนาได้เป็นส่วน ๆ แล้วให้โปรแกรมแต่ละส่วนนั้นร่วมกันใช้งานหน่วยความจำที่มีอยู่ด้วยหลักการเดียวกันกับหน่วยความจำเสมือน (Virtual Memory) ซึ่งเรียกวิธีการนี้ว่า "โอเวอร์เลย์ (Overlay)" การทำโอเวอร์เลย์เป็นขั้นตอนหนึ่งที่ต้องทำในระหว่างการรวมโมดูลเป็นโปรแกรม ซึ่งในชุดพัฒนาโปรแกรมคลิปเปอร์ก็ได้เตรียมโปรแกรม RTLINK สำหรับทำหน้าที่ลิงค์เอาไว้ให้ โดยมีขีดความสามารถในการสร้างโอเวอร์เลย์ได้ด้วย

ในบทความนี้เป็นการอธิบายถึงขั้นตอนการทำโอเวอร์เลย์ให้กับโปรแกรมที่พัฒนาด้วยคลิปเปอร์ ทั้งโอเวอร์เลย์แบบสแตติก (Static Overlay) และไดนามิก (Dynamic Overlay) รวมถึงวิธีการสร้างไฟล์แยกเก็บโอเวอร์เลย์ (External Overlay)

การใช้งาน RTLINK

ก่อนที่จะกล่าวถึงวิธีการสร้างการโอเวอร์เลย์จะขอแนะนำถึงการใช้งานโปรแกรม RTLINK เสียก่อน ซึ่งการใช้งานโปรแกรม RTLINK นั้นมีหลายรูปแบบแต่ในที่นี้จะเพียงการแนะวิธีใช้งาน

เบื้องต้นโดยจะอธิบายถึงการใช้งานในรูปแบบอิสระ (FREE FORMAT) ซึ่งเป็นรูปแบบที่มีอยู่เดิมในโปรแกรมลิงค์ PLINK86 ของคลิปเปอร์รุ่นซัมเมอร์ 86

รูปแบบการใช้งาน (FREE FORMAT)

```
RTLINK [FILE <objFILE List>]
       [OUTPUT <outputFILE>]
       [LIBRARY <libFILE List>]
        List>]
        [<Link Option List>]
```

การเรียกใช้งาน RTLINK มี 3 วิธีคือ

1. เรียกใช้งานในบรรทัดเดียว (Command Line) คือการเรียกโปรแกรมแล้วตามด้วยรายการที่ต้องการ เช่น

```
RTLINK FI TEST OUT FIRST
```

รายการที่ใส่จะใส่ได้ไม่เกิน 128 ตัวอักษร (คำสั่งที่ใช้ระบุชื่อออปเจ็กต์-ไฟล์ ไลบรารีไฟล์และไฟล์ผลลัพธ์ สามารถย่อได้ เช่น FILE ย่อเป็น FI เป็นต้น)

2. แบบตอบโต้ (Prompt Mode) โดยเรียกโปรแกรม RTLINK แล้วกดแป้นพิมพ์ Enter โดยเมื่อ RTLINK ทำงานจะแสดงเครื่องหมายเตรียมพร้อม "=>" แล้วรอรับรายการจากผู้ใช้ ซึ่งต้องป้อนรายละเอียดต่าง ๆ ที่ต้องการลงไป เช่น

RTLINK กด ENTER

.RTLINK for Clipper Dynamic Overlay linker/ (C) Copyright.

=> FILE TEST กด ENTER

=> OUTPUT FIRST กด ENTER

=>; กด ENTER

3. สั่งงานด้วยไฟล์สคริปต์ (Script File Mode) สร้างไฟล์ข้อความเก็บรายการการสั่งงาน RTLINK เอาไว้ในไฟล์ดังกล่าวโดยมีนามสกุล .LNK เช่น ไฟล์ TEST.LNK

```
# Test.lnk-ILink
FILE TEST
OUTPUT FIRST
```

เครื่องหมาย '#' คือคำอธิบาย (comment)

โอเวอร์เลย์ (Overlay)

โปรแกรมที่มีขนาดใหญ่เกินหน่วยความจำที่เหลือของเครื่องจะไม่สามารถทำงานได้ เพราะเป็นข้อจำกัดของระบบดำเนินการดอส (DOS) เองที่สามารถใช้งานหน่วยความจำของเครื่องได้เพียง 640 กิโลไบต์เท่านั้น ตามปกติเมื่อมีการเรียกโปรแกรมขึ้นมาทำงานบนดอส หน้าที่ของดอสจะต้องจัดเตรียมหน่วยความจำขนาดตามความต้องการของโปรแกรมที่ถูกเรียกทำงาน ซึ่งหากหน่วยความจำที่เหลือของเครื่องมีไม่พอ ดังนั้นโปรแกรมที่ถูกเรียกดังกล่าวก็จะไม่สามารถทำงานบนเครื่องได้ ดังตัวอย่างในรูปที่ 1 สมมติว่าโปรแกรมซึ่งมีขนาด 620 กิโลไบต์และถูกเรียกมาทำงานหากบนเครื่องขณะนี้นั้นมีหน่วยความจำในเครื่องเหลืออยู่เพียง 590 กิโลไบต์ โปรแกรมดังกล่าวก็ไม่สามารถจะทำงานได้

การแก้ไข

หากเครื่องคอมพิวเตอร์ที่ใช้ระบบดำเนินการแบบหลายงาน (Multi-Tasking) ซึ่งมีการจัดการหน่วยความจำแบบหน่วยความจำเสมือน (Virtual Memory) ในกรณีนี้ระบบดำเนินการจะหยุดโปรแกรมใดโปรแกรมหนึ่งที่ทำงานอยู่ชั่วคราวแล้วจัดเก็บโปรแกรมนั้นลงดิสก์เสียก่อนเพื่อเตรียมพื้นที่หน่วยความจำว่างให้แก่โปรแกรมที่จะถูกเรียกทำงาน

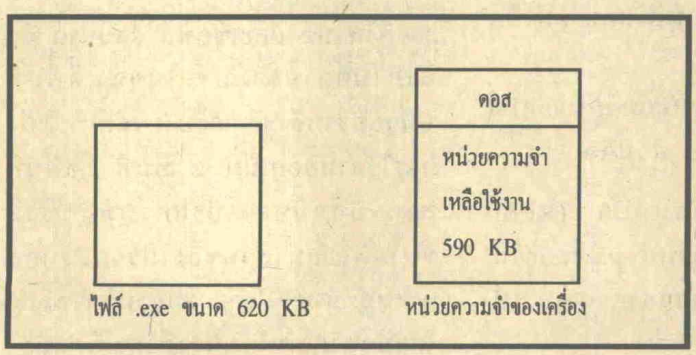
แต่ในระบบดำเนินการดอสซึ่งเป็นระบบดำเนินการแบบงานเดียวไม่สามารถจะทำเช่นนั้นได้ การแก้ไขต้องทำที่ตัวโปรแกรมเองโดยการแบ่งโปรแกรมออกเป็น 2 ส่วนดังนี้

- ส่วนโปรแกรมที่ต้องใช้ประจำจะถูกโหลดค้างเอาไว้ในหน่วยความจำตลอดเวลา

- ส่วนโปรแกรมที่ถูกเรียกใช้ในบางครั้งก็จะโหลดลงหน่วยความจำเมื่อต้องการ โปรแกรมส่วนนี้มักจะมีด้วยกันหลายส่วนย่อย

เช่นเดียวกันในการใช้งานหน่วยความจำ โปรแกรมก็จะมองหน่วยความจำเป็น 2 พื้นที่คือ

- พื้นที่หน่วยความจำหลัก (Root Area) เพื่อใช้สำหรับโหลดโปรแกรม



รูปที่ 1 แสดงหน่วยความจำที่เหลือเพียง 590 กิโลไบต์โดยโปรแกรมจะไม่สามารถทำงานได้

ส่วนแรกที่ต้องใช้งานประจำ

- พื้นที่หน่วยความจำที่ใช้งานร่วมกัน (Sharing Area) สำหรับโหลดโปรแกรมส่วนที่ 2 ลงเมื่อถูกเรียกใช้งานสลับกันไปมา

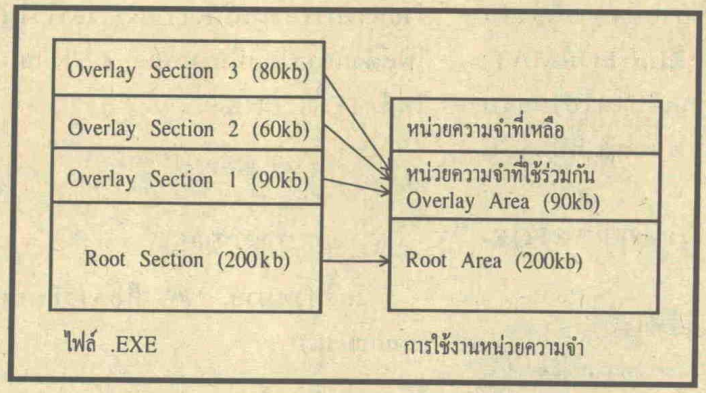
การทำโอเวอร์เลย์มีอยู่ 2 วิธีคือ โอเวอร์เลย์แบบสแตติก (Static Overlay) และไดนามิก (Dynamic Overlay) มีความแตกต่างกันในด้านของการเขียนโปรแกรมและการใช้งานหน่วยความจำที่ต้องใช้ร่วมกัน ซึ่งการจัดโอเวอร์เลย์นี้โปรแกรมลิงเกอร์จะเป็นตัวจัดการให้ในขั้นตอนของการรวมโปรแกรม (Link) และโปรแกรมลิงเกอร์จะนำเอาไลบรารีซึ่งมีโปรแกรมจัดการโอเวอร์เลย์มารวมไว้ในโปรแกรมที่จะสร้างขึ้นด้วย เพื่อคอยควบคุมการใช้งานหน่วยความจำให้เป็นไปตามที่กำหนดในขั้นตอนของการลิงค์ โดยมีรายละเอียดการทำงานและวิธีการสร้างดังนี้

โอเวอร์เลย์แบบสแตติก (Static Overlay)

โอเวอร์เลย์แบบสแตติกมีมาแต่เดิมในคลิปเปอร์รุ่นซัมเมอร์ 86 (PKLINK-86) ซึ่งในคลิปเปอร์รุ่น 5.X (5.0, 5.01 5.2) ก็ยังคงเอาไว้อยู่ โดยเตรียมไว้สำหรับโมดูลที่เขียนด้วยภาษาอื่นที่ไม่ใช่คลิปเปอร์ เช่น ซี แอสเซมบลี เป็นต้น ที่ไม่สามารถทำโอเวอร์เลย์แบบไดนามิกได้

โอเวอร์เลย์แบบสแตติกนี้จะแบ่งโปรแกรมออกเป็น 2 ส่วนคือ

- โปรแกรมส่วนหลัก (Root Section) ซึ่งโปรแกรมส่วนนี้จะอยู่ในหน่วยความจำตลอดเวลาตั้งแต่แรกที่โปรแกรมถูกเรียกทำงาน



รูปที่ 2
ขนาดของไฟล์ .EXE ที่แบ่งเป็น 2 ส่วน คือส่วนโปรแกรมหลักและส่วนเซกชัน

```
BEGINAREA
SECTION FILE ชื่อไฟล์ออปเจ็กต์.... ชื่อไฟล์ออปเจ็กต์
.
.
SECTION FILE ชื่อไฟล์ออปเจ็กต์..... ชื่อไฟล์ออปเจ็กต์
ENDAREA
```

รูปที่ 3
โครงสร้างของโมดูลที่เป็นโอเวอร์เลย์

- โปรแกรมส่วนโอเวอร์เลย์ (Overlay Section) ซึ่งโปรแกรมส่วนนี้จะถูกโหลดลงหน่วยความจำก็ต่อเมื่อมีการเรียกใช้ โดยจะลงในหน่วยความจำบริเวณพื้นที่ที่จัดให้ใช้งานร่วมกัน

ดังตัวอย่างในรูปที่ 2 ขนาดของโปรแกรมในไฟล์ .EXE เท่ากับ 430 กิโลไบต์ แบ่งโปรแกรมออกเป็น 2 ส่วน คือส่วนของโปรแกรมหลัก (Root Section) ซึ่งมีขนาด 200 กิโลไบต์ และอีกส่วนคือส่วนของโอเวอร์เลย์ซึ่งประกอบด้วยส่วนย่อย 3 ส่วนด้วยกัน คือเซกชัน 1 ขนาด 90 กิโลไบต์ เซกชัน 2 ขนาด 60 กิโลไบต์ และเซกชัน 3 ขนาด 80 กิโลไบต์ตามลำดับ ในขณะที่พื้นที่หน่วยความจำของเครื่องก็จะมีการแบ่งการใช้งานออกเป็น 2 พื้นที่ คือพื้นที่สำหรับส่วนของโปรแกรมหลักซึ่งมีขนาดพื้นที่เท่าส่วนของโปรแกรมหลัก และพื้นที่สำหรับโปรแกรมส่วนโอเวอร์เลย์โดยมีขนาดเท่ากับขนาดที่ใหญ่ที่สุดของโปรแกรมโอเวอร์เลย์ส่วนย่อยคือ 90 กิโลไบต์นั่นเอง ซึ่งจะทำให้โปรแกรมโอเวอร์เลย์ส่วนย่อยทุกส่วนถูกโหลดลงในพื้นที่นี้ได้

การสร้างโอเวอร์เลย์แบบสแตติก

โอเวอร์เลย์แบบสแตติกนี้ผู้พัฒนาโปรแกรมจะช่วยโปรแกรมลิงเกอร์ในการแบ่งโปรแกรม ซึ่งวิธีการทำนั้นไม่ยุ่งยากมาก เพียงแค่เขียนคำสั่งเพื่อแบ่งส่วนของโปรแกรมลงในไฟล์สคริปต์ของลิงเกอร์ (ไฟล์นามสกุล .LNK) ดังนี้

- โมดูลที่ต้องอยู่ในส่วนหลักให้ใส่ไว้หลังคำสั่ง FILE

- โมดูลใดที่เป็นส่วนโอเวอร์เลย์ให้ใส่ไว้ในแต่ละส่วนย่อยซึ่งมีโครงสร้างดังรูปที่ 3

ตัวอย่าง

สมมติว่าโปรแกรมคลิปเปอร์โปรแกรมหนึ่งประกอบด้วยโมดูลต่อไปนี้คือ main, sub1, sub1a, sub1b, sub2, sub3 และ sub4 โดยโมดูล main เป็นโมดูลหลักและโมดูลที่เหลือเป็นโมดูลย่อยซึ่งมีขนาดของแต่ละโมดูลดังตารางที่ 1

จัดแบ่งโปรแกรมออกเป็นส่วนโอเวอร์เลย์ดังนี้

- ส่วนหลักคือโมดูล main
- ส่วนโอเวอร์เลย์แบ่งเป็นส่วนย่อยดังนี้

ส่วนที่ 1 ประกอบด้วยโมดูล sub1, sub1a, sub1b รวมกันขนาดของส่วนนี้คือ 80 กิโลไบต์

ส่วนที่ 2 มีโมดูล sub2 ขนาดของส่วนนี้คือ 70 กิโลไบต์

ส่วนที่ 3 มีโมดูล sub3 ขนาดของส่วนนี้คือ 90 กิโลไบต์

ส่วนที่ 4 มีโมดูล sub4 ขนาดของส่วนนี้คือ 100 กิโลไบต์

ซึ่งสามารถเขียนไฟล์สคริปต์ของลิงเกอร์ได้ดังรูปที่ 4

ผลการสร้างโปรแกรมจากสคริปต์จะได้ดังรูปที่ 5

ข้อพิจารณาในการสร้างโอเวอร์เลย์แบบสแตติก

1. โปรแกรมที่ต้องถูกเรียกใช้งานประจำ ควรจัดเอาไว้ในส่วนหลัก (Root Area) โดยหากนำเอาไว้ในส่วนโอเวอร์เลย์จะทำให้โปรแกรมทำงานช้า เพราะจะถูกโหลดเข้าออกระหว่างดิสค์และหน่วยความจำบ่อย เช่น นอกเหนือจากโมดูลหลักคือ main แล้วหากมีโมดูลย่อยอื่นอีก เช่น โมดูลสำหรับจัดการระบบหน้าต่างและเมนู ซึ่งโมดูลเหล่านี้จะถูกใช้งานประจำจึงควรจัดเอาไว้ในส่วนหลัก

2. การแบ่งกลุ่มโมดูลหรือออปเจ็กต์ในส่วนโอเวอร์เลย์ย่อย ที่จะใช้พื้นที่หน่วยความจำร่วมกันแต่ละส่วน (Section) ควรให้มีขนาดใกล้เคียงกันเพื่อใช้หน่วยความจำอย่างมีประสิทธิภาพ เช่น ในตัวอย่างแต่ละส่วนโอเวอร์เลย์ย่อยจะมีขนาด 80, 70, 90 และ 100 ตาม

ชื่อโมดูล	ขนาด (กิโลไบต์)
main	200
sub1	50
sub1a	20
sub1b	10
sub2	70
sub3	90
sub4	100

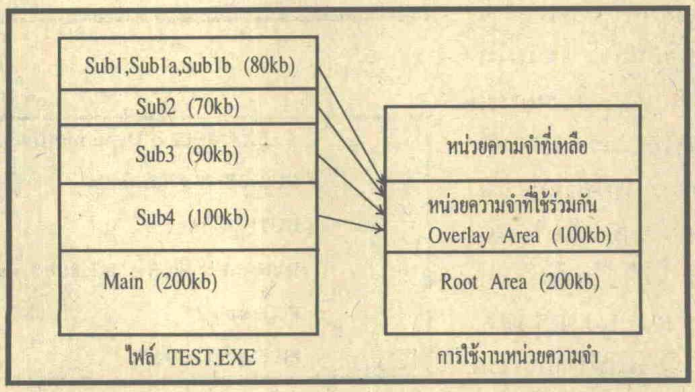
ตารางที่ 1 แสดงขนาดของโมดูลหลักและโมดูลย่อย

ลำดับซึ่งจะเห็นว่าขนาดใกล้เคียงกัน โดยในกรณีนี้จะทำให้เกิดพื้นที่หน่วยความจำที่ใช้งานร่วมกันขนาด 100 กิโลไบต์

3. โปรแกรมในแต่ละส่วนโอเวอร์เลย์ย่อย (Section) ใด ๆ จะต้องไม่มีการเรียกใช้งานฟังก์ชัน โพรซีเดอร์ หรือตัวแปรในส่วนโอเวอร์เลย์ย่อยอื่นที่ใช้งานหน่วยความจำร่วมกัน เพราะจะทำให้โปรแกรมไม่สามารถทำงานต่อไปได้ เนื่องจากต้องนำส่วนโอเวอร์เลย์ย่อย 2 ส่วนที่มีการเรียกใช้งานร่วมกันมาลงในพื้นที่หน่วยความจำที่เดียวกันและพร้อม ๆ กันซึ่งเป็นไปไม่ได้ ซึ่งหากจำเป็นที่จะต้องกระทำเช่นนี้ ก็จะต้องสร้างพื้นที่หน่วยความจำขึ้นมาอีกพื้นที่หนึ่งด้วยการประกาศ BEGINAREA ...ENDAREA อีกส่วนหนึ่ง

```
#EXAMPLE1.LNK- Example Static Overlay Link File
OUT test
FILE main # Root Area
BEGINAREA # Overlay Section
        SECTION FILE sub1,sub1a,sub1b
        SECTION FILE sub2
        SECTION FILE sub3
        SECTION FILE sub4
ENDAREA
```

รูปที่ 4 ไฟล์สคริปต์ของลิงค์เกอร์ในการสร้างโอเวอร์เลย์แบบสแตติก



รูปที่ 5 ผลการสร้างโปรแกรมจากสคริปต์ของโอเวอร์เลย์แบบสแตติก

```
#EXAMPLE2.LNK -     Example Static Overlay Link File
FILE main
OUTPUT test
BEGINAEREA
    BEGINAREA
        SECTION FILE sub1
        SECTION FILE sub1a
        SECTION FILE sub1b
    ENDAREA
    SECTION PRELOAD FILE sub2
    SECTION FILE sub3
    SECTION FILE sub4
ENDAREA
```

รูปที่ 6
ตัวอย่างไฟล์สคริปต์ที่มีการสร้างพื้นที่หน่วยความจำที่ใช้ร่วมกัน 2 พื้นที่

ตัวอย่างที่ซับซ้อน

ตัวอย่างไฟล์สคริปต์ต่อไปนี้จะมีการสร้างพื้นที่หน่วยความจำที่ใช้ร่วมกัน 2 พื้นที่โดยพื้นที่ที่สองซ้อนอยู่ในพื้นที่แรก นอกจากนี้สังเกตในเซกชันของ sub2 มีคำสั่งพิเศษคือ PRELOAD คำสั่งนี้จะมีผลทำให้มีการโหลดโมดูล sub2 ก่อนที่จะมีการทำงานของโปรแกรม ซึ่งตามปกติแล้วส่วนหลักของโปรแกรมเท่านั้นที่จะถูกโหลดก่อนที่โปรแกรมจะทำงาน (ดูรูปที่ 6)

ไลบรารีจัดการโอเวอร์เลย์แบบสแตติก

เมื่อได้มีการกำหนดให้มีการสร้างโอเวอร์เลย์แบบสแตติกแล้ว ดังนั้นในระหว่างการลิงค์โปรแกรม โปรแกรม RTLINK จะนำเอาไลบรารี RTLUTILS.LIB เข้ามารวมกับโปรแกรมที่จะสร้างขึ้นด้วย ซึ่งการนำมารวมให้นั้นเป็นไปโดยอัตโนมัติ ผู้ใช้ไม่ต้องระบุคำสั่งใด ๆ เลย ไฟล์ RTLUTILS.LIB เป็นไลบรารีที่จำเป็นต้องมีให้โปรแกรมทุกโปรแกรมที่มีโอเวอร์เลย์แบบสแตติก เพราะเป็นไลบรารีที่ทำหน้าที่จัดการโอเวอร์เลย์แบบสแตติกโดยเฉพาะ

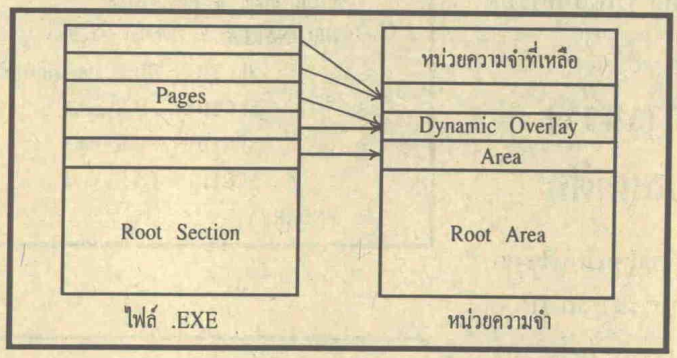
โอเวอร์เลย์แบบไดนามิก (Dynamic Overlay)

การลิงค์โปรแกรมในคลิปเปอร์ 5.X โปรแกรม RTLINK จะสร้างโอเวอร์เลย์แบบไดนามิกให้กับทุกโมดูลที่เขียนด้วยคลิปเปอร์โดยอัตโนมัติ (สำหรับโมดูลที่เขียนด้วยภาษาอื่นไม่สามารถทำโอเวอร์เลย์แบบไดนามิกได้ หากต้องการทำโอเวอร์เลย์ต้องทำเป็นโอเวอร์เลย์แบบสแตติกเท่านั้น) การทำงานของโอเวอร์เลย์แบบไดนามิกมีดังนี้คือ

แบ่งโปรแกรมออกเป็น 2 ส่วน ดังนี้

- โปรแกรมส่วนหลัก (Root Section) คือส่วนของโปรแกรมที่จะอยู่ในหน่วยความจำตลอด (เช่นเดียวกับโอเวอร์เลย์แบบสแตติก

- ส่วนที่เหลือของโปรแกรมจะถูกแบ่งออกเป็นชิ้นเล็ก ๆ (Page) โดยในหน่วยความจำก็จะถูกแบ่งออกเป็นบริเวณเล็ก ๆ ขนาดเท่ากัน (แต่จำนวนจะน้อยกว่าชิ้นของโปรแกรม) แต่ละชิ้นของโปรแกรมจะถูกโหลดลงหน่วยความจำต่อเมื่อมีการเรียกใช้ การโหลด



รูปที่ 7
แสดงการทำงานของโอเวอร์เลย์แบบไดนามิก

```
# EXAMPLE3 Dynamic Overlay Link File
RESIDENT FILE main
OUTPUT test
DYNAMIC FILE sub1,sub1a,sub1b
FILE sub2
FILE sub3,sub4
```

รูปที่ 8
ตัวอย่างไฟล์สคริปต์ของโอเวอร์เลย์แบบไดนามิก

นั้นจะโหลดลงในบริเวณเล็กที่เตรียมไว้ให้ แต่หากบริเวณเหล่านั้นไม่ว่างก็จะโหลดทับลงในบริเวณที่ไม่ค่อยถูกเรียกใช้งาน โดยใช้วิธีการแทนที่แบบ LRU (LRU ย่อมาจาก Least recently used) ดูรูปที่ 7

ตัวอย่าง

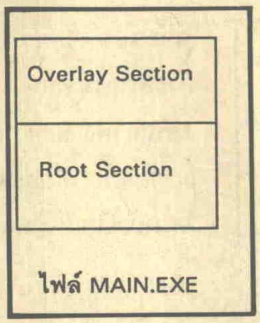
จากตัวอย่างโปรแกรมที่ผ่านมาในเรื่องโอเวอร์เลย์แบบสแตติก หากเขียนไฟล์สคริปต์ใหม่ดังรูปที่ 8

ตามปกติแล้วโมดูลทุกโมดูลที่เขียนด้วยคลิปเปอร์นั้น RTLINK จะลิงค์เป็นโอเวอร์เลย์แบบไดนามิกให้โดยอัตโนมัติอยู่แล้ว ไม่จำเป็นต้องใช้คำสั่ง DYNAMIC อีก ดังนั้นในการกำหนดโมดูล sub1, sub1a, sub1b ไม่จำเป็นต้องสั่ง DYNAMIC ก็ได้ แต่ถ้าหากว่าเราต้องการจะวางโมดูลบางโมดูลเอาไว้ที่พื้นที่หลักนอกเหนือโมดูลที่เขียนด้วยซีหรือแอสเซมบลีก็สามารถทำได้โดยใช้คำสั่ง RESIDENT ดังตัวอย่างที่ต้องการให้โมดูล main อยู่ที่พื้นที่หลักเป็นต้น

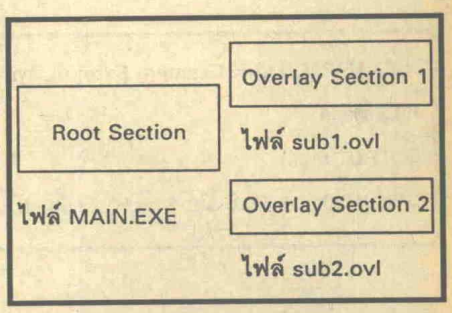
การแยกส่วนโอเวอร์เลย์เก็บเป็นไฟล์ (External Overlay)

ตามปกติส่วนของโปรแกรมที่ถูกแบ่งออกเป็นโอเวอร์เลย์มักจะยังคงรวมอยู่ในไฟล์ผลลัพธ์เดียวกันที่มีขนาดใหญ่ อย่างไรก็ตาม หากต้องการจะแยกส่วนโปรแกรมที่เป็นของโอเวอร์เลย์เก็บเอาไว้คนละไฟล์แยกจากไฟล์โปรแกรมผลลัพธ์ก็จะทำได้ และทำให้เกิดความสะดวกต่อการจัดเก็บ และสามารถจัดเก็บโปรแกรมลงแผ่นดิสค์ที่มีขนาดจำกัดได้

ตัวอย่างเช่น หากโปรแกรมคลิปเปอร์ที่พัฒนาได้มีขนาดใหญ่กว่าแผ่นฟลอปปี้ดิสค์ที่จะต้องจัดเก็บ และมีความจำเป็นต้องบรรจุโปรแกรมลงแผ่นดิสค์



รูปที่ 9 จัดเก็บโอเวอร์เลย์รวมกันกับโปรแกรมส่วนหลัก



รูปที่ 10 จัดเก็บโอเวอร์เลย์แยกกันกับโปรแกรมส่วนหลัก

```
#EXAMPLE4.LNK -     Example External Static Overlay Link File
FILE main    # Root Area
OUTPUT test
BEGINAREA   # Overlay Section
        SECTION INTO sub1.ovl FILE sub1,sub1a,sub1b
        SECTION INTO sub1.ovl FILE sub2
        SECTION INTO sub2.ovl FILE sub3
        SECTION INTO sub2.ovl FILE sub4
ENDAREA
```

รูปที่ 11 แสดงการแยกส่วนโอเวอร์เลย์แบบสแตติกเก็บเป็นไฟล์

ให้ได้ ซึ่งเมื่อจัดเก็บแล้วจะต้องเรียกใช้งานจากแผ่นดิสค์ได้อีกด้วย ในกรณีนี้หากมีการแยกโปรแกรมออกเป็นไฟล์ โดยส่วนหลักของโปรแกรมได้เป็นไฟล์ main.exe ส่วนของโอเวอร์เลย์ได้ 2 ไฟล์คือ sub1.ovl และ sub2.ovl และจัดเก็บด้วยการแยกเอาไฟล์ main.exe เอาไว้ในแผ่นดิสค์หนึ่ง แล้วแยกไฟล์โอเวอร์เลย์ที่เหลือทั้งสองไฟล์เก็บลงในแผ่นดิสค์แผ่นที่สอง ก็สามารถจัดลงในแผ่นดิสค์ได้พอดี

การเรียกใช้งานโปรแกรมที่จัดเก็บดังกล่าวก็สามารถทำได้อย่างไม่มีปัญหา โดยเริ่มต้นจากแผ่นแรกที่เก็บไฟล์ main.exe การเรียกโปรแกรม main.exe จะทำให้เกิดการโหลดเอาส่วนหลักของโปรแกรมไปไว้ในหน่วยความจำและค้างเอาไว้ ดังนั้นแผ่นดิสค์แผ่นนี้ไม่ต้องใช้งานอีกแล้วให้นำออกจากเครื่องแล้วนำแผ่นที่สองมาใส่ไว้แทน ซึ่งแผ่นที่สองนี้จะต้องค้างเอาไว้ในเครื่องตลอดเวลา เพราะขณะใช้งานจะมีการอ่านเขียนไฟล์โอเวอร์เลย์ตลอดเวลา

รูปที่ 9 เป็นการจัดเก็บโอเวอร์เลย์รวมกันกับโปรแกรมส่วนหลัก ส่วนรูปที่ 10 เป็นการจัดเก็บโอเวอร์เลย์แยกกันกับโปรแกรมส่วนหลัก

การแยกส่วนโอเวอร์เลย์แบบสแตติกเก็บเป็นไฟล์ (External Static Overlay)

การแยกส่วนโอเวอร์เลย์แบบสแตติกเก็บเป็นไฟล์ทำได้โดยการเปลี่ยนคำสั่งจาก SECTION FILE <ชื่อโมดูล,...> เป็น SECTION INTO <ชื่อไฟล์โอเวอร์เลย์> FILE <ชื่อโมดูล,...>

ตัวอย่าง (ดูรูปที่ 11)

```
# EXAMPLE5.LNK Example Externa Dynamic Overlay Link File
FILE main
OUTPUT main
DYN INTO sub FILE sub1,sub1a,sub1b,sub2,sub3,sub4
```

รูปที่ 12 ตัวอย่างการเขียนสคริปต์ไฟล์แสดงการแยกส่วนโอเวอร์เลย์แบบไดนามิกเก็บเป็นไฟล์

ตัวอย่างต่อไปนี้เป็นการแยกเก็บส่วนโอเวอร์เลย์เป็นไฟล์โดยมีโมดูล sub1, sub1a, sub1b และ sub2 เก็บอยู่ในไฟล์ sub1.ovl ส่วนโมดูล sub3 และ sub4 เก็บอยู่ในไฟล์ sub2.ovl

การแยกส่วนโอเวอร์เลย์แบบไดนามิกเก็บเป็นไฟล์ (External Dynamic Overlay)

ให้ใช้คำสั่ง DYN INTO <ชื่อไฟล์โอเวอร์เลย์> FILE <ชื่อโมดูล,...>

ตัวอย่าง

หากต้องการแยกเก็บส่วนโอเวอร์เลย์ทั้งหมดรวมกันในไฟล์ sub สามารถทำได้ดังนี้

กรณีเรียกใช้งานในบรรทัดเดียว (Command Line)

```
RTLINK FILE main OUTPUT main DYN INTO sub FILE sub1, sub1a, sub1b, sub2, sub3, sub4
```

กรณีเขียนเป็นสคริปต์ไฟล์ (ดูรูปที่ 12)

Prelinked Libraries

Prelinked Libraries ในคลิปเปอร์ 5.2 คือไฟล์ BASE52.PLL เป็นที่รวมของไลบรารีและโมดูลออปเจ็กต์ต่าง ๆ เอาไว้ในไฟล์โดยบรรจุข้อมูล (ข้อมูลที่สำคัญคือข้อมูล FIXUP สำหรับใช้ในการค้นหาชื่อตัวแปร ฟังก์ชัน โพรซีเดอร์ที่จะถูกเรียกใช้งานจากภายนอก) ที่จำเป็นเตรียมพร้อมสำหรับโหลดลงหน่วยความจำเพื่อทำงาน

Prelinked Libraries จะช่วยลดขนาดและเวลาของไฟล์โปรแกรม .EXE ที่จะได้จากการลิงค์ ขนาดที่ลดลงในไฟล์โปรแกรม .EXE คือโมดูลต่าง ๆ ที่ถูกนำไปเก็บรวมอยู่ในไฟล์ BASE52.PLL โดยไฟล์นี้จะถูกใช้งานด้วยการโหลดลงไปในหน่วยความจำเมื่อมีการเรียกใช้งานไฟล์โปรแกรม .EXE ซึ่งในลักษณะนี้หากมีไฟล์โปรแกรม .EXE หลาย ๆ โปรแกรมก็สามารถจะใช้ไฟล์ BASE52.PLL ร่วมกันได้ ซึ่งจะทำให้ประหยัดพื้นที่ดิสค์ในการจัดเก็บ

ตารางที่ 2 นี้เปรียบเทียบการสร้างโปรแกรมด้วยวิธีธรรมดากับการใช้ Prelinked Libraries จะเห็นว่ากรณีนี้การใช้ Prelinked Libraries จะใช้พื้นที่ดิสค์มากกว่าธรรมดาถึง 23 กิโลไบต์

จากตัวอย่างข้างต้นอาจยังไม่เห็นถึงการประหยัดพื้นที่ดิสค์ แต่ตัวอย่างต่อไปนี้จะช่วยให้เข้าใจ โดยสมมติว่าหากมีการพัฒนาโปรแกรมคลิปเปอร์ขึ้นมาชุดหนึ่ง ซึ่งต้องแยกโปรแกรมออกเป็นโมดูลอิสระในรูปของไฟล์โปรแกรม .EXE เพื่อใช้งานในแต่ละแผนกดังนี้ โปรแกรมบัญชีทั่วไป โปรแกรมบัญชีรายรับ โปรแกรมบัญชีรายจ่าย โปรแกรมรับสั่งสินค้า และโปรแกรมสินค้าคงคลัง การพัฒนาโปรแกรมใช้ Prelinked Libraries ช่วยโดยติดตั้งโปรแกรมลงในระบบเครือข่ายแลน (LAN) เพื่อให้เข้าใจง่ายขึ้นจะสมมติว่าโมดูลแต่ละส่วนมีขนาดเท่ากัน และแสดงตารางเปรียบเทียบกับการสร้างโปรแกรมแบบเดิมกับการสร้างโปรแกรมด้วย Prelinked Libraries โดยพิจารณาถึงจำนวนโปรแกรมที่เพิ่มขึ้นดังนี้ (ดูตารางที่ 3)

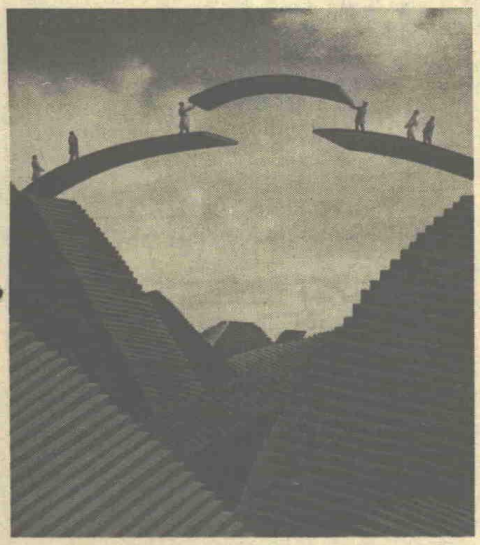
โปรแกรมอิสระต่าง ๆ จะร่วมกันใช้งานไฟล์ BASE52.PLL ดังรูปที่ 13

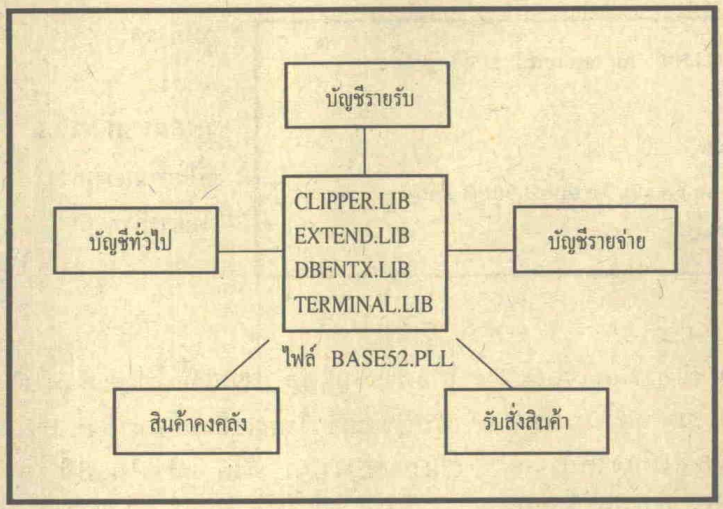
โปรแกรม .EXE อิสระ		.PLL และ .EXE	
ชื่อไฟล์	ขนาด	ชื่อไฟล์	ขนาด
EXAMPLE.EXE	327KB	BASE52.PLL	345KB
		EXAMPLE.EXE	5KB

ตารางที่ 2 เปรียบเทียบการสร้างโปรแกรมด้วยวิธีธรรมดากับการใช้ Prelinked Libraries

จำนวนไฟล์โปรแกรม	โปรแกรม .EXE อิสระ	.PLL และ .EXE	ประหยัด
1 ไฟล์	327K	345 + 5 = 350	– 23
2 ไฟล์	2*327 = 654	345 + 2*5 = 355	299
3 ไฟล์	3*327 = 981	345 + 3*5 = 360	621
4 ไฟล์	4*327 = 1308	345 + 4*5 = 365	943
5 ไฟล์	5*327 = 1635	345 + 5*5 = 370	1265

ตารางที่ 3 แสดงการเปรียบเทียบการสร้างโปรแกรมแบบเดิมกับการสร้างโปรแกรมด้วย Prelinked Libraries





รูปที่ 13
โปรแกรมอิสระจะร่วมกันใช้งานไฟล์

สรุปประโยชน์ของ Prelinked Libraries

- ลดพื้นที่ในการเก็บโปรแกรม .EXE ไลบรารีหรือโมดูลที่ใช้ร่วมกันให้ใช้ในไฟล์ PLL
- ทำให้การลิงค์เร็วขึ้น

การสร้าง BASE52.PLL

สร้างจากไฟล์สคริปต์ BASE52.LNK ที่คลิปเปอร์เตรียมเอาไว้ให้

```
RTLINK @BASE52
```

และจะต้องสร้าง DOS ENVIRONMENT เพื่อให้โปรแกรม .EXE สามารถเรียกใช้ .PLL ได้

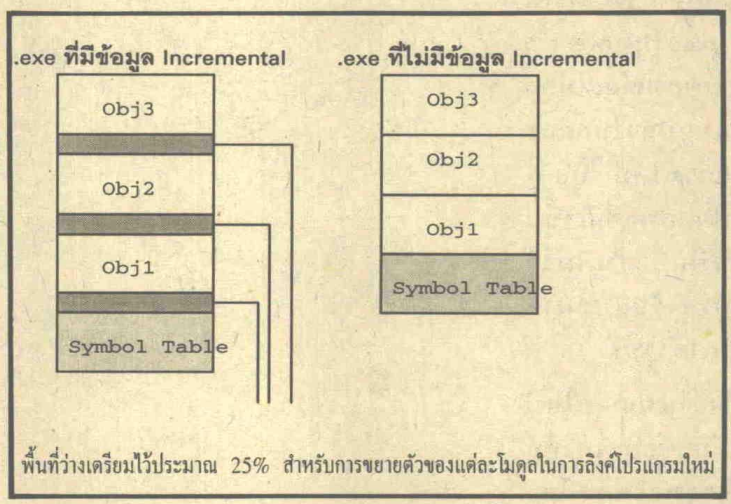
```
SET PLL=C:\CLIPPER5\PLL
```

และในการลิงค์โปรแกรมเพื่อทำ Prelinked Libraries นั้นเพียงแค่เพิ่มคำสั่ง PLL <ชื่อ Prelinked Libraries> ซึ่งขณะลิงค์โปรแกรม RTLINK ต้องการไฟล์ .PLT ด้วย ซึ่งไฟล์นี้จะถูกสร้างขึ้นพร้อมกับการสร้าง .PLL

ตัวอย่าง (ดูรูปที่ 14)

```
# EXAMPLE6.LNK - Link Script file for TEST.EXE
FILE test
OUTPUT test
LIB CLIPPER, EXTEND, TERMINAL, DBFNTX
PLL BASE52
```

รูปที่ 14
ตัวอย่างไฟล์สคริปต์ของโปรแกรมที่ใช้ Prelinked Libraries



รูปที่ 15
พื้นที่ว่างเตรียมไว้ประมาณ 25 เปอร์เซ็นต์สำหรับการขยายตัวของโมดูลในการลิงค์โปรแกรม

การเพิ่มความเร็วในการลิงค์ (Incremental Linking)

การพัฒนาโปรแกรมด้วยคลิปเปอร์นั้น ขั้นตอนการลิงค์โปรแกรมเป็นขั้นตอนที่ใช้เวลามากที่สุดขั้นตอนหนึ่ง เพราะโปรแกรมลิงเกอร์จะต้องรวมโมดูลทั้งหมดเข้าด้วยกันและคำนวณตำแหน่งของตัวแปร ฟังก์ชัน โพรซีเดอร์ที่มีการเรียกใช้งานใหม่เสมอ แต่จริง ๆ แล้ว

```
#EXAMPLE6.LNK - Incremental Linking
FILE main
OUTPUT main
FILE sub1,sub1a,sub1b,sub2,sub3,sub4
INCREMENTAL
```

รูปที่ 16 คำสั่ง INCREMENTAL ที่ใช้เพิ่มความเร็วในการลิงค์

การพัฒนาโปรแกรมจะมีการแก้ไขและเปลี่ยนแปลงเพียงไม่กี่โมดูล ดังนั้นหากสามารถลดขั้นตอนการลิงค์ลงได้บ้างก็จะทำให้การพัฒนาโปรแกรมเสร็จโดยเร็วขึ้น (การลดเวลาการแปลโปรแกรมทำได้ด้วยโปรแกรม RMAKE ซึ่งได้กล่าวไปแล้ว) ในโปรแกรม RTLINK ได้เตรียมหนทางที่ใช้ช่วยในการประหยัดเวลาการลิงค์ลงได้ด้วยวิธีที่เรียกว่า Incremental Linking โดยวิธีการนี้โปรแกรม RTLINK จะสร้างไฟล์นามสกุล .INF โดยใช้ชื่อเดียวกับโปรแกรมผลลัพธ์ ในไฟล์นี้ RTLINK จะใช้เก็บข้อมูลคือรายชื่อโมดูลต่าง ๆ พร้อมทั้งเวลาและขนาดของแต่ละโมดูลเพื่อใช้เทียบการเปลี่ยนแปลงในแต่ละครั้งของการลิงค์โปรแกรมใหม่ นอกจากนั้นที่ตัวโปรแกรมผลลัพธ์ที่เกิดขึ้น RTLINK จะเตรียมพื้นที่ว่างเพิ่มไว้ในแต่ละโมดูลสำหรับรองรับกับขนาดของโมดูลที่เปลี่ยนแปลงดังรูปที่ 15

เมื่อใดที่มีการลิงค์โปรแกรมใหม่ RTLINK จะเปรียบเทียบข้อมูลในไฟล์ .INF กับไฟล์ออปเจ็กต์ทุกไฟล์ โดยหากไฟล์ใดที่ไม่มีการเปลี่ยนแปลงหรือเปลี่ยนแปลงเล็กน้อยโดยขนาดที่เพิ่มไม่เกิน 25 เปอร์เซ็นต์ที่เตรียมไว้การลิงค์จะเป็นไปอย่างรวดเร็ว แต่หากมีการเปลี่ยนแปลงมากก็จะเกิดลิงค์อย่างเต็มที่ อย่างไรก็ตาม หากต้องการลิงค์แบบเต็มที่ใหม่ทุก ๆ โมดูลก็สามารถทำได้โดยการลบไฟล์ .INF ทิ้งไป คำสั่งที่ใช้เพิ่มความเร็วในการลิงค์คือคำสั่ง INCREMENTAL ดังตัวอย่างในรูปที่ 16

ส่งท้าย

ข้อเสียของการเพิ่มความเร็วในการลิงค์คือ ทำให้ไฟล์ผลลัพธ์มีขนาดใหญ่ ซึ่งระหว่างการแก้ไขปรับปรุงโปรแกรมคงไม่เป็นไร แต่เมื่อใดที่เสร็จสิ้นการพัฒนาแล้วอาจจะสั่งลิงค์โดยไม่เพิ่มความเร็วการลิงค์เพื่อจะได้ขนาดของโปรแกรมผลลัพธ์เล็กลง

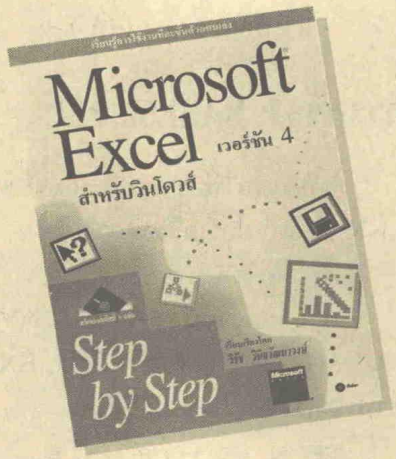
ซีเอ็ด

ใหม่ล่าสุด

Microsoft Excel เวอร์ชัน 4 สำหรับวินโดวส์

พร้อมแผ่นดิสก์ 1 แผ่น

ภายในเล่มเราจะได้เรียนรู้เกี่ยวกับ :
- การใช้ทูลบาร์สำหรับงานทั่วไปที่ใช้กันบ่อย ๆ
- การสร้างและแก้ไขชาร์ต
- สร้าง แก้ไข และจัดรูปแบบเวอร์กชีต
- การจัดระบบและวิเคราะห์ดาต้าเบส
- เชื่อมโยงเวอร์กชีตเพื่อการปรับปรุงอัตโนมัติ
- การสร้างแมคโครเพื่อช่วยประหยัดเวลา
- กระจัดระบบและทำงานกับหลาย ๆ เวอร์กชีต
- การสร้างชุดข้อมูลสำหรับแทนค่าในเวอร์กชีต
- แปลงเวอร์กชีต แมคโคร และรูปแบบของโลตัส 1-2-3



หาซื้อได้ตามร้านหนังสือชั้นนำทั่วไป
หรือที่ ซีเอ็ดบุ๊คเซ็นเตอร์ทุกสาขา
และสั่งซื้อได้ที่ซีเอ็ด สำนักงานใหญ่
โทร 248-6280

มีจำหน่ายแล้ว